Creating a modern Angular frontend web application for controlling rental cars involves several steps, including setting up the project structure, implementing routing, creating models, integrating MDB Bootstrap for forms, and interacting with API CRUD endpoints. Below is a step-by-step guide with command lines, folder structures, and source code examples.

### Prompt GHAT GPT

Create a modern and real world frontend webapplication for controlling rental cars in Angular and use MDB Bootstrap for the forms, and considering API CRUD Endpoints ( ***List, Create, Update and Delete*** ). Provide a complete command lines and full folders structures. Additionaly provide full source code example for each file considering routing, models, and app.models.ts \_\_\_

### Step 1: Setup Angular Project

# Create a new Angular project  
ng new rental-car-management  
  
# Change directory to the project folder  
cd rental-car-management  
  
# Install MDB Bootstrap  
npm install mdb-angular-ui-kit

### Step 2: Create Folder Structure

rental-car-management  
│  
└── src  
 ├── app  
 │ ├── models  
 │ │ └── car.model.ts  
 │ ├── services  
 │ │ └── car.service.ts  
 │ ├── components  
 │ │ ├── car-list  
 │ │ │ ├── car-list.component.html  
 │ │ │ ├── car-list.component.ts  
 │ │ │ └── car-list.component.css  
 │ │ ├── car-create  
 │ │ │ ├── car-create.component.html  
 │ │ │ ├── car-create.component.ts  
 │ │ │ └── car-create.component.css  
 │ │ ├── car-edit  
 │ │ │ ├── car-edit.component.html  
 │ │ │ ├── car-edit.component.ts  
 │ │ │ └── car-edit.component.css  
 │ │ └── car-details  
 │ │ ├── car-details.component.html  
 │ │ ├── car-details.component.ts  
 │ │ └── car-details.component.css  
 │ ├── app-routing.module.ts  
 │ └── app.module.ts  
 └── assets

### Step 3: Define Models

**src/app/models/car.model.ts**

export interface Car {  
 id: number;  
 brand: string;  
 model: string;  
 year: number;  
 color: string;  
}

### Step 4: Implement Services

**src/app/services/car.service.ts**

import { Injectable } from '@angular/core';  
import { HttpClient } from '@angular/common/http';  
import { Observable } from 'rxjs';  
import { Car } from '../models/car.model';  
  
@Injectable({  
 providedIn: 'root'  
})  
export class CarService {  
 private apiUrl = 'api/cars'; // Replace with your actual API endpoint  
  
 constructor(private http: HttpClient) {}  
  
 getCars(): Observable<Car[]> {  
 return this.http.get<Car[]>(this.apiUrl);  
 }  
  
 createCar(car: Car): Observable<Car> {  
 return this.http.post<Car>(this.apiUrl, car);  
 }  
  
 updateCar(car: Car): Observable<Car> {  
 const url = `${this.apiUrl}/${car.id}`;  
 return this.http.put<Car>(url, car);  
 }  
  
 deleteCar(id: number): Observable<void> {  
 const url = `${this.apiUrl}/${id}`;  
 return this.http.delete<void>(url);  
 }  
}

### Step 5: Create Components

#### Car List Component

**src/app/components/car-list/car-list.component.ts**

import { Component, OnInit } from '@angular/core';  
import { Car } from '../../models/car.model';  
import { CarService } from '../../services/car.service';  
  
@Component({  
 selector: 'app-car-list',  
 templateUrl: './car-list.component.html',  
 styleUrls: ['./car-list.component.css']  
})  
export class CarListComponent implements OnInit {  
 cars: Car[];  
  
 constructor(private carService: CarService) { }  
  
 ngOnInit(): void {  
 this.carService.getCars().subscribe(cars => this.cars = cars);  
 }  
}

**src/app/components/car-list/car-list.component.html**

<!-- Car list HTML template goes here -->

#### Car Create Component

**src/app/components/car-create/car-create.component.ts**

import { Component } from '@angular/core';  
import { Car } from '../../models/car.model';  
import { CarService } from '../../services/car.service';  
  
@Component({  
 selector: 'app-car-create',  
 templateUrl: './car-create.component.html',  
 styleUrls: ['./car-create.component.css']  
})  
export class CarCreateComponent {  
 car: Car = {  
 id: null,  
 brand: '',  
 model: '',  
 year: null,  
 color: ''  
 };  
  
 constructor(private carService: CarService) { }  
  
 createCar(): void {  
 this.carService.createCar(this.car).subscribe();  
 }  
}

**src/app/components/car-create/car-create.component.html**

<!-- Car create HTML template goes here -->

#### Car Edit Component

**src/app/components/car-edit/car-edit.component.ts**

import { Component, OnInit } from '@angular/core';  
import { ActivatedRoute } from '@angular/router';  
import { Car } from '../../models/car.model';  
import { CarService } from '../../services/car.service';  
  
@Component({  
 selector: 'app-car-edit',  
 templateUrl: './car-edit.component.html',  
 styleUrls: ['./car-edit.component.css']  
})  
export class CarEditComponent implements OnInit {  
 car: Car;  
  
 constructor(  
 private route: ActivatedRoute,  
 private carService: CarService  
 ) { }  
  
 ngOnInit(): void {  
 const id = +this.route.snapshot.paramMap.get('id');  
 this.carService.getCar(id).subscribe(car => this.car = car);  
 }  
  
 updateCar(): void {  
 this.carService.updateCar(this.car).subscribe();  
 }  
}

**src/app/components/car-edit/car-edit.component.html**

<!-- Car edit HTML template goes here -->

#### Car Details Component

**src/app/components/car-details/car-details.component.ts**

import { Component, OnInit } from '@angular/core';  
import { ActivatedRoute } from '@angular/router';  
import { Car } from '../../models/car.model';  
import { CarService } from '../../services/car.service';  
  
@Component({  
 selector: 'app-car-details',  
 templateUrl: './car-details.component.html',  
 styleUrls: ['./car-details.component.css']  
})  
export class CarDetailsComponent implements OnInit {  
 car: Car;  
  
 constructor(  
 private route: ActivatedRoute,  
 private carService: CarService  
 ) { }  
  
 ngOnInit(): void {  
 const id = +this.route.snapshot.paramMap.get('id');  
 this.carService.getCar(id).subscribe(car => this.car = car);  
 }  
  
 deleteCar(): void {  
 this.carService.deleteCar(this.car.id).subscribe();  
 }  
}

**src/app/components/car-details/car-details.component.html**

<!-- Car details HTML template goes here -->

### Step 6: Define Routing

**src/app/app-routing.module.ts**

import { NgModule } from '@angular/core';  
import { Routes, RouterModule } from '@angular/router';  
import { CarListComponent } from './components/car-list/car-list.component';  
import { CarCreateComponent } from './components/car-create/car-create.component';  
import { CarEditComponent } from './components/car-edit/car-edit.component';  
import { CarDetailsComponent } from './components/car-details/car-details.component';  
  
const routes: Routes = [  
 { path: '', redirectTo: '/cars', pathMatch: 'full' },  
 { path:  
  
 'cars', component: CarListComponent },  
 { path: 'cars/create', component: CarCreateComponent },  
 { path: 'cars/:id/edit', component: CarEditComponent },  
 { path: 'cars/:id', component: CarDetailsComponent }  
];  
  
@NgModule({  
 imports: [RouterModule.forRoot(routes)],  
 exports: [RouterModule]  
})  
export class AppRoutingModule { }

### Step 7: Update App Module

**src/app/app.module.ts**

import { NgModule } from '@angular/core';  
import { BrowserModule } from '@angular/platform-browser';  
import { HttpClientModule } from '@angular/common/http';  
import { AppRoutingModule } from './app-routing.module';  
import { AppComponent } from './app.component';  
import { CarListComponent } from './components/car-list/car-list.component';  
import { CarCreateComponent } from './components/car-create/car-create.component';  
import { CarEditComponent } from './components/car-edit/car-edit.component';  
import { CarDetailsComponent } from './components/car-details/car-details.component';  
  
@NgModule({  
 declarations: [  
 AppComponent,  
 CarListComponent,  
 CarCreateComponent,  
 CarEditComponent,  
 CarDetailsComponent  
 ],  
 imports: [  
 BrowserModule,  
 AppRoutingModule,  
 HttpClientModule  
 ],  
 providers: [],  
 bootstrap: [AppComponent]  
})  
export class AppModule { }

### Step 8: Usage

You would need to replace the API endpoints (apiUrl) in the car.service.ts file with your actual API endpoints.

### Conclusion

This is a basic setup for a modern Angular frontend web application for controlling rental cars, utilizing MDB Bootstrap for forms and integrating API CRUD endpoints. You can further enhance the application by adding authentication, error handling, and additional features as per your requirements.